<<

[Bitwise Operators]

Description

The left shift operator << causes the bits of the left operand to be shifted **left** by the number of positions specified by the right operand.

Syntax

variable << number\_of\_bits;

Parameters

variable: Allowed data types: byte, int, long.  
number\_of\_bits: a number that is < = 32. Allowed data types: int.

Example Code

int a = 5; // binary: 0000000000000101

int b = a << 3; // binary: 0000000000101000, or 40 in decimal

Notes and Warnings

When you shift a value x by y bits (x << y), the leftmost y bits in x are lost, literally shifted out of existence:

int x = 5; // binary: 0000000000000101

int y = 14;

int result = x << y; // binary: 0100000000000000 - the first 1 in 101 was discarded

If you are certain that none of the ones in a value are being shifted into oblivion, a simple way to think of the left-shift operator is that it multiplies the left operand by 2 raised to the right operand power. For example, to generate powers of 2, the following expressions can be employed:

Operation Result

--------- ------

1 << 0 1

1 << 1 2

1 << 2 4

1 << 3 8

...

1 << 8 256

1 << 9 512

1 << 10 1024

...

The following example can be used to print out the value of a received byte to the serial monitor, using the left shift operator to move along the byte from bottom(LSB) to top (MSB), and print out its Binary value:

// Prints out Binary value (1 or 0) of byte

void printOut1(int c) {

for (int bits = 7; bits > -1; bits--) {

// Compare bits 7-0 in byte

if (c & (1 << bits)) {

Serial.print("1");

}

else {

Serial.print("0");

}

}

}